09/981954 Deepak Metha

EAST SEARCH

6/10/05

L#	Hits	Search String	Databases
S1	2	6,282,131.pn.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S2	14	(memory near2 compiler$1) with (memory near2 instance$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S3	0	(memory near2 compiler$1) with charcaterization	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S4	10	(memory near2 instance$1) with compilable	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S5	81	(memory near2 instance$1) with (parameter$1 or parametric)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S6	1628	(memory near2 instance$1) with ((data near2 point$1) or data)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S7	22	(memory near2 compiler$1) with (parameter$1 or parametric)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S8	198	(memory near2 compiler$1) with ((data near2 point$1) or data)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S9	1880	S2 or S4 or S5 or S6 or S7 or S8	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S10	1	S9 and (memory with (MUX near2 factor$1))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S11	1	S9 and (MUX near2 factor$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S12	2	S9 and (memory with ((parametric near2 dataset$1) or dataset$1))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S13	0	S9 and (congruent near2 (memory near2 instance$1))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S14	0	S9 and (congruent with (memory near2 instance$1))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S15	16	S9 and (scale near2 factor$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S16	0	S9 and ((scale near2 factor$1) near2 interpolat$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S17	109	S9 and (memory near2 timing)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S18	250	S9 and (memory with ((access or cycle) near2 time))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S19	37	S17 and S18	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S20	0	S9 and (MUX-4 or MUX-8 or MUX-16 or MUX-32)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S21	176	S9 and ((memory near2 instance$1) with (ROM or ((static or dynamic) near2 RAM) or EPROM	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S22	4	S9 and ((scale near2 factor$1) with interpolat$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S23	22	S17 and S21	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S24	14	S18 and S21	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S25	5	S9 and ((memory near2 compiler$1) with simulat$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S26	4	S9 and ((memory near2 compiler$1) with technolog$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S27	1	S9 and ((memory with technolog$3) with ("1.0" or "0.8" or "0.6" or "0.2"))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S28	44	S9 and ((design near2 rule$1) or foundry-specific or rule-specific or process-specific or proces	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S29	150	S10 or S11 or S12 or S15 or S19 or S22 or S23 or S24 or S25 or S26 or S27 or S28 or S2 or	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S30	2	(memory near2 compiler$1) with characterization	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S31	14	(memory near2 compiler$1) with (memory near2 instance$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S32	10	(memory near2 instance$1) with compilable	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S33	81	(memory near2 instance$1) with (parameter$1 or parametric)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S34	1628	(memory near2 instance$1) with ((data near2 point$1) or data)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S35	22	(memory near2 compiler$1) with (parameter$1 or parametric)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S36	198	(memory near2 compiler$1) with ((data near2 point$1) or data)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB

S37	1880	S31 or S32 or S33 or S34 or S35 or S36	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S38	1	S37 and (memory with (MUX near2 factor$1))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S39	1	S37 and (MUX near2 factor$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S40	2	S37 and (memory with ((parametric near2 dataset$1) or dataset$1))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S41	16	S37 and (scale near2 factor$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S42	109	S37 and (memory near2 timing)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S43	250	S37 and (memory with ((access or cycle) near2 time))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S44	37	S42 and S43	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S45	176	S37 and ((memory near2 instance$1) with (ROM or ((static or dynamic) near2 RAM) or EPROI	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S46	4	S37 and ((scale near2 factor$1) with interpolat$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S47	22	S42 and S45	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S48	14	S43 and S45	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S49	5	S37 and ((memory near2 compiler$1) with simulat$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S50	4	S37 and ((memory near2 compiler$1) with technolog$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S51	1	S37 and ((memory with technolog$3) with ("1.0" or "0.8" or "0.6" or "0.2"))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S52	44	S37 and ((design near2 rule$1) or foundry-specific or rule-specific or process-specific or proce	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S54	6324	(memory near2 characteriz$5)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S55	44	S54 and (memory near2 compiler$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S56	248	S54 and (memory near2 instance$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S57	286	S55 or S56	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S58	1	S37 and (multiplex$3 near2 factor$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S59	2	S57 and (multiplex$3 near2 factor$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S60	37	S37 and (memory with ("1.0" or "0.8" or "0.6" or "0.2"))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S61	18	S57 and (memory with ("1.0" or "0.8" or "0.6" or "0.2"))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S62	6	S55 and S56	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S63	98	S55 or S58 or S59 or S60 or S61 or S62	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB
S53	150	S38 or S39 or S40 or S41 or S44 or S46 or S47 or S48 or S49 or S50 or S51 or S52 or S31 o	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB

09/981954 **Deepak Metha**

EAST SEARCH

6/10/05

Results of search set S91:

S60 and (receiver with demodulator)

Document Kind Codes	Title	Issue Date	Current OR	Abstract
US 20050114560 A1	Tightly coupled and scalable memory and execution unit architecture	20050526	710/22	
US 20050108398 A1	Systems and methods for using metrics to control throttling and swapping in a message proce:	20050519	709/225	
US 20050102472 A1	Data processor having cache memory	20050512	711/120	
US 20050060500 A1	General purpose memory compiler system and associated methods	20050317	711/147	
US 20050055675 A1	Generation of software objects from a hardware description	20050310	717/135	

US 20050047238 A1	Reconfigurable memory arrays	20050303	365/210
US 20050039156 A1	Design method for essentially digital systems and components thereof and essentially digital sy	20050217	716/18
US 20040215893 A1	Method for use of ternary cam to implement software programmable cache policies	20041028	711/138
US 20040205290 A1	Memory device	20041014	711/103
US 20040202025 A1	Nonvolatile semiconductor memory device	20041014	365/185.29
US 20040202019 A1	Nonvolatile semiconductor memory device	20041014	365/185.01
US 20040196712 A1	Semiconductor memory device	20041007	365/202
US 20040151038 A1	Memory module and method for operating a memory module in a data memory system	20040805	365/200
US 20040122644 A1	Optimized execution of software objects generated from a hardware description	20040624	703/16
US 20040117168 A1	Global analysis of software objects generated from a hardware description	20040617	703/14
US 20040117167 A1	Simulation of software objects generated from a hardware description	20040617	703/14
US 20040111690 A1	Method for composing memory on programmable platform devices to meet varied memory re	20040610	716/17
US 20040088702 A1	Lock-free overflow strategy for work stealing	20040506	718/100
US 20040071009 A1	Compilable address magnitude comparator for memory array self-testing	20040415	365/145
US 20040036700 A1	Data communications device, data communications system, document display method with vid	20040226	345/660
US 20040015925 A1	Method, article of manufacture and apparatus for performing automatic intemodule call linkage	20040122	717/155
US 20030225740 A1	Memory model for a run-time environment	20031204	707/1
US 20030204676 A1	Data processor having cache memory	20031030	711/137
US 20030192013 A1	Method and apparatus for facilitating process-compliant layout optimization	20031009	716/2
US 20030178648 A1	Gate array core cell for VLSI ASIC devices	20030925	257/202
US 20030156751 A1	Method of and apparatus for rectifying a stereoscopic image	20030821	382/154
US 20030105772 A1	Write-barrier maintenance in a garbage collector	20030605	707/103R
US 20030103379 A1	Semiconductor memory device	20030605	365/185.2
US 20030026131 A1	Redundancy circuit and method for replacing defective memory cells in a flash memory device	20030206	365/185.11
US 20030026129 A1	REDUNDANCY CIRCUIT AND METHOD FOR FLASH MEMORY DEVICES	20030206	365/185.09
US 20030002347 A1	Nonvolatile semiconductor memory device	20030102	365/185.29
US 20020191448 A1	Timing circuit and method for a compilable dram	20021219	365/194
US 20020176282 A1	Nonvolatile semiconductor memory device	20021128	365/185.22
US 20020154553 A1	System and method for redundancy implementation in a semiconductor device	20021024	365/200
US 20020131320 A1	SRAM emulator	20020919	365/233
US 20020083262 A1	MEMORY DEVICE OPERABLE WITH A SMALL-CAPACITY BUFFER MEMORY AND HAVIN	20020627	711/103
US 20020046251 A1	Streaming memory controller	20020418	709/213
US 20020042897 A1	Method and system for distributed testing of electronic devices	20020411	714/718
US 20020035671 A1	Processor with cache control	20020321	711/118
US 20020013881 A1	Dynamically-tunable memory controller	20020131	711/105
US 20010048610 A1	Semiconductor memory device	20011206	365/185.2
US 20010037432 A1	Data processor having cache memory	20011101	711/129
US 20010030889 A1	Nonvolatile semiconductor memory device	20011018	365/185.05
US 20010014933 A1	Memory management table producing method and memory device	20010816	711/154
US 6895452 B1	Tightly coupled and scalable memory and execution unit architecture	20050517	710/22
US 6892328 B2	Method and system for distributed testing of electronic devices	20050510	714/42
US 6853572 B1	Methods and apparatuses for a ROM memory array having twisted source or bit lines	20050208	365/63
US 6850446 B1	Memory cell sensing with low noise generation	20050201	365/206

US 6848027 B2	Data processor having cache memory	20050125	711/129
US 6842375 B1	Methods and apparatuses for maintaining information stored in a non-volatile memory cell	20050111	365/185.18
US 6836831 B2	Independent sequencers in a DRAM control structure	20041228	711/169
US 6791882 B2	Nonvolatile semiconductor memory device	20040914	365/185.29
US 6788574 B1	Electrically-alterable non-volatile memory cell	20040907	365/185.08
US 6765245 B2	Gate array core cell for VLSI ASIC devices	20040720	257/202
US 6747902 B2	Nonvolatile semiconductor memory apparatus	20040608	365/185.29
US 6745372 B2	Method and apparatus for facilitating process-compliant layout optimization	20040601	716/2
US 6738953 B1	System and method for memory characterization	20040518	716/1
US 6735120 B2	Semiconductor device having a high-speed data read operation	20040511	365/185.2
US 6711092 B1	Semiconductor memory with multiple timing loops	20040323	365/233
US 6704834 B1	Memory with vectorial access	20040309	711/5
US 6678643 B1	Event based semiconductor test system	20040113	703/14
US 6658610 B1	Compilable address magnitude comparator for memory array self-testing	20031202	714/718
US 6647465 B2	Realtime parallel processor system for transferring common information among parallel proces	20031111	711/131
US 6625712 B2	Memory management table producing method and memory device	20030923	711/202
US 6598190 B1	Memory device generator for generating memory devices with redundancy	20030722	714/711
US 6597629 B1	Built-in precision shutdown apparatus for effectuating self-referenced access timing scheme	20030722	365/233
US 6594177 B2	Redundancy circuit and method for replacing defective memory cells in a flash memory device	20030715	365/185.11
US 6587927 B2	Data processor having cache memory	20030701	711/129
US 6587364 B1	System and method for increasing performance in a compilable read-only memory (ROM)	20030701	365/63
US 6584036 B2	SRAM emulator	20030624	365/233
US 6578129 B1	Optimized virtual memory management for dynamic data types	20030610	711/209
US 6563732 B2	Redundancy circuit and method for flash memory devices	20030513	365/185.09
US 6556490 B2	System and method for redundancy implementation in a semiconductor device	20030429	365/200
US 6538932 B2	Timing circuit and method for a compilable DRAM	20030325	365/194
US 6532174 B2	Semiconductor memory device having high speed data read operation	20030311	365/185.2
US 6473356 B1	Low power read circuitry for a memory circuit based on charge redistribution between bitlines a	20021029	365/230.03
US 6466504 B1	Compilable block clear mechanism on per I/O basis for high-speed memory	20021015	365/218
US 6453434 B2	Dynamically-tunable memory controller	20020917	714/718
US 6438670 B1	Memory controller with programmable delay counter for tuning performance based on timing p	20020820	711/167
US 6438036 B2	Nonvolatile semiconductor memory device	20020820	365/185.22
US 6434658 B1	Memory device operable with a small-capacity buffer memory and having a flash memory	20020813	711/103
US 6425116 B1	Automated design of digital signal processing integrated circuit	20020723	716/18
US 6425062 B1	Controlling burst sequence in synchronous memories	20020723	711/167
US 6424556 B1	System and method for increasing performance in a compilable read-only memory (ROM)	20020723	365/63
US 6405160 B1	Memory compiler interface and methodology	20020611	703/24
US 6370078 B1	Way to compensate the effect of coupling between bitlines in a multi-port memories	20020409	365/230.05
US 6363020 B1	Architecture with multi-instance redundancy implementation	20020326	365/200
US 6356503 B1	Reduced latency row selection circuit and method	20020312	365/230.06
US 6348774 B1	Method for controlling several stepping motor modules with prior loading of ramp data	20020219	318/685
US 6334174 B1	Dynamically-tunable memory controller	20011225	711/167
US 6292427 B1	Hierarchical sense amp and write driver circuitry for compilable memory	20010918	365/230.03

US 6282131 B1	Self-timed clock circuitry in a multi-bank memory instance using a common timing synchronizat	20010828	365/191
US 6275902 B1	Data processor with variable types of cache memories and a controller for selecting a cache m	20010814	711/129
US 6259629 B1	Nonvolatile semiconductor memory device	20010710	365/185.22
US 6249901 B1	Memory characterization system	20010619	716/5
US 6249471 B1	Fast full signal differential output path circuit for high-speed memory	20010619	365/207
US 6236618 B1	Centrally decoded divided wordline (DWL) memory architecture	20010522	365/230.06
US 6233197 B1	Multi-port semiconductor memory and compiler having capacitance compensation	20010515	365/230.05
US 6216180 B1	Method and apparatus for a nonvolatile memory interface for burst read operations	20010410	710/35
US 6181600 B1	Nonvolatile semiconductor memory device	20010130	365/185.18
US 6157576 A	Nonvolatile semiconductor memory device	20001205	365/185.29
US 6016273 A	Nonvolatile semiconductor memory device	20000118	365/185.22
US 6000522 A	Multi-compartment and acceptors computerized vending machine	19991214	194/217
US 5991200 A	Nonvolatile semiconductor memory device	19991123	365/185.18
US 5970986 A	Apparatus for rejection diagnostics after organ transplants	19991026	128/899
US 5970240 A	Method and apparatus for configurable memory emulation	19991019	703/25
US 5959894 A	Nonvolatile semiconductor memory device	19990928	365/185.29
US 5949715 A	Nonvolatile semiconductor memory device	19990907	365/185.22
US 5923610 A	Timing scheme for memory arrays	19990713	365/230.08
US 5917752 A	Nonvolatile semiconductor memory device	19990629	365/185.18
US 5848432 A	Data processor with variable types of cache memories	19981208	711/131
US 5844842 A	Nonvolatile semiconductor memory device	19981201	365/185.24
US 5808900 A	Memory having direct strap connection to power supply	19980915	716/10
US 5781732 A	Framework for constructing shared documents that can be collaboratively accessed by multipl	19980714	709/205
US 5781476 A	Nonvolatile semiconductor memory device	19980714	365/185.22
US 5644753 A	Fast, dual ported cache controller for data processors in a packet switched cache coherent mu	19970701	711/131
US 5640349 A	Flash memory system	19970617	365/185.33
US 5634107 A	Data processor and method of processing data in parallel	19970527	711/111
US 5555555 A	Apparatus which detects lines approximating an image by repeatedly narrowing an area of the	19960910	382/104
US 5528552 A	Dynamic random access memory device with sense amplifiers serving as cache memory inde	19960618	365/238.5
US 5493507 A	Digital circuit design assist system for designing hardware units and software units in a desired	19960220	703/14
US 5479374 A	Semiconductor memory device employing sense amplifier control circuit and word line control c	19951226	365/233.5
US 5479184 A	Videotex terminal system using CRT display and binary-type LCD display	19951226	345/3.1
US 5452226 A	Rule structure for insertion of new elements in a circuit design synthesis procedure	19950919	716/18
US 5400267 A	Local in-device memory feature for electrically powered medical equipment	19950321	702/59
US 5399912 A	Hold-type latch circuit with increased margin in the feedback timing and a memory device usin	19950321	327/94
US 5222029 A	Bitwise implementation mechanism for a circuit design synthesis procedure	19930622	716/18
US 5175707 A	Semiconductor memory device having a driving circuit provided in association with a high spee	19921229	365/230.06
US 5050091 A	Integrated electric design system with automatic constraint satisfaction	19910917	716/10
US 5046113 A	Method of and apparatus for detecting pattern defects by means of a plurality of inspecting unit	19910903	382/147
US 4945495 A	Image memory write control apparatus and texture mapping apparatus	19900731	345/552
US 4875192 A	Semiconductor memory with an improved nibble mode arrangement	19891017	365/193
US 4845640 A	High-speed dual mode graphics memory	19890704	345/572
US 4803476 A	Video terminal for use in graphics and alphanumeric applications	19890207	345/545

US 4688182 A	Method and apparatus for generating a set of signals representing a curve	19870818	345/442	
US 4686636 A	Method and apparatus for generating a set of signals representing a curve	19870811	345/442	
US 4686634 A	Method and apparatus for generating a set of signals representing a curve	19870811	345/442	
US 4686633 A	Method and apparatus for generating a set of signals representing a curve	19870811	345/442	
US 4465349 A	Microfilm card and a microfilm reader with automatic stage positioning	19840814	353/25	
US 4431007 A	Referenced real-time ultrasonic image display	19840214	600/440	
US 4314331 A	Cache unit information replacement apparatus	19820202	711/133	
US 4245304 A	Cache arrangement utilizing a split cycle mode of operation	19810113	711/122	
US 4208716 A	Cache arrangement for performing simultaneous read/write operations	19800617	711/3	
EP 647900 A1	Parameter storage space allocation.	19950412		
US 20050060500 A	Memory compiler units accessing method for generating memory related design files, involves	20050317		
US 6738953 B	Memory e.g. ROM characterization method, involves creating hierarchically-stitched parametric	20040518		
US 6282131 B	Timing synchronization method in memory instance, involves enabling address signals for subs	20010828		15
US 6249901 B	Automatic memory characterization for designing integrated circuit, involves simulating circuit b	20010619		43
EP 191134 A	Display data encoding of signals representing curve - using parametric cubic polynomial functic	19860820		
EP 175179 A	Signal generation method for points on curve - using compiler in form of Hermite cubic parame	19860326		

IEEE Xplore® RELEASE 2.0

Home | Login | Logout | Access Information | Alerts |

Search Results BROWSE SEARCH IEEE XPLORE GUIDE

Results for **"(('memory compiler' and simulat*)<in>metadata)"**
Your search matched **4** of **1168854** documents.
A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance** in **Descending** order.

e-mail

» View Session History

» New Search

Modify Search

(('memory compiler' and simulat*)<in>metadata) »

□ Check to search only within this results set

Display Format: ● Citation ○ Citation & Abstract

» Key

▨	Indicates full text access
IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

Select Article Information

□ 1. **A widely configurable EPROM memory compiler for embedded applications**
Lim, H.; Shubat, A.; Duvalyan, V.; Dandamudi, S.; Raviv, S.; Kablanian, A.;
Memory Technology, Design and Testing, 1998. Proceedings. International Workshop
24-25 Aug. 1998 Page(s):12 - 16
Abstract | Full Text: PDF(40 KB) IEEE CNF

□ 2. **Evaluating the impact of architectural-level optimizations on clock power**
Duarte, D.; Narayanan, V.; Irwin, M.J.; Kandemir, M.;
ASIC/SOC Conference, 2001. Proceedings. 14th Annual IEEE International
12-15 Sept. 2001 Page(s):447 - 451
Abstract | Full Text: PDF(464 KB) IEEE CNF

□ 3. **A diffused CMOS SRAM compiler for gate-arrays**
Gee, P.; Tou, J.;
Circuits and Systems, 1991., Proceedings of the 34th Midwest Symposium on
14-17 May 1991 Page(s):807 - 810 vol.2
Abstract | Full Text: PDF(304 KB) IEEE CNF

□ 4. **A novel memory architecture for real-time mesh-based video motion compensati**
Sayed, M.; Badawy, W.;
System-on-Chip for Real-Time Applications, 2004.Proceedings. 4th IEEE International
19-21 July 2004 Page(s):153 - 157
Abstract | Full Text: PDF(280 KB) IEEE CNF

Help Contact Us Privacy & :

Indexed by Inspec®

IEEE Xplore® RELEASE 2.0

Home | Login | Logout | Access Information | Alerts |

Welcome United States Patent and Trademark Office

Search Results

BROWSE SEARCH IEEE XPLORE GUIDE

Results for "(('memory characterization')<in>metadata)" e-mail
Your search matched **6** of **1168854** documents.
A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance** in **Descending** order.

» View Session History

» New Search

Modify Search

(('memory characterization')<in>metadata) »

□ Check to search only within this results set

Display Format: ● Citation ○ Citation & Abstract

» **Key**

IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

Select Article Information

□ 1. **PASTEL: a parameterized memory characterization system**
Ogawa, K.; Kohno, M.; Kitamura, F.;
Design, Automation and Test in Europe, 1998., Proceedings
23-26 Feb. 1998 Page(s):15 - 20
AbstractPlus | Full Text: PDF(192 KB) IEEE CNF

□ 2. **Memory characterization of SiGe quantum dot flash memories with HfO/sub 2/ ar tunneling dielectrics**
Dong-Won Kim; Taehoon Kim; Banerjee, S.K.;
Electron Devices, IEEE Transactions on
Volume 50, Issue 9, Sept. 2003 Page(s):1823 - 1829
AbstractPlus | References | Full Text: PDF(566 KB) IEEE JNL

□ 3. **Memory characterization of a parallel data mining workload**
Jin-Soo Kim; Xiaohan Qin; Yarsun Hsu;
Workload Characterization: Methodology and Case Studies, 1998
29 Nov. 1998 Page(s):60 - 68
AbstractPlus | Full Text: PDF(156 KB) IEEE CNF

□ 4. **Workload Characterization: Methodology and Case Studies. Based on the First W Workload Characterization**
Workload Characterization: Methodology and Case Studies, 1998
29 Nov. 1998
AbstractPlus | Full Text: PDF(108 KB) IEEE CNF

□ 5. **A Programmable Time Measurement Architecture for Embedded Memory Charac**
Collins, M.; Al-Hashimi, B.M.; Ross, N.;
Test Symposium, 2005. European
22-25 May 2005 Page(s):128 - 133
AbstractPlus | Full Text: PDF(176 KB) IEEE CNF

□ 6. **2003 IEEE International Workshop on Workload Characterization (IEEE Cat. No.0**
Workload Characterization, 2003. WWC-6. 2003 IEEE International Workshop on
27 Oct. 2003
AbstractPlus | Full Text: PDF(181 KB) IEEE CNF

PORTAL
USPTO

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ● The ACM Digital Library ○ The Guide

+"memory compiler" +simulat*

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used **memory compiler simulat**

Found **57** of **156,259**

Sort results by: publication date

Display results: expanded form

Save results to a Binder

Search Tips

□ Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Does not have memory compiler

Results 1 - 20 of 57

Result page: **1** 2 3 next

Relevance scale

1 Integration, Verification and Layout of a Complex Multimedia SOC

Chien-Liang Chen, Jiing-Yuan Lin, Youn-Long Lin

March 2005 **Proceedings of the conference on Design, Automation and Test in Europe - Volume 2**

Full text available: pdf(76.49 KB) Additional Information: full citation, abstract

We present our experience of designing a single-chip controller for advanced digital still camera from specification all the way to mass production. The process involves collaboration with camera system designer, IP vendors, EDA vendors, silicon wafer foundry, package & testing houses, and camera maker. We also co-work with academic research groups to develop a JPEG codec IP and memory BIST and SOC testing methodology. In this presentation, we cover the problems encountered, our solutions, and l ...

2 Using Mobilize Power Management IP for Dynamic & Static Power Reduction in SoC at 130 nm

Dan Hillman

March 2005 **Proceedings of the conference on Design, Automation and Test in Europe - Volume 3**

Full text available: pdf(149.06 KB) Additional Information: full citation, abstract

At 130 nm and 90 nm, power consumption (both dynamic and static) has become a barrier in the roadmap for SoC designs targeting battery powered, mobile applications. This paper presents the results of dynamic and static power reduction achieved implementing Tensilica's 32-bit Xtensa microprocessor core, using Virtual Silicon's Power Management IP. Independent voltage islands are created using Virtual Silicon's VIP PowerSaver standard cells by using voltage level shifting cells and voltage isolati ...

3 A way-halting cache for low-energy high-performance systems

Chuanjun Zhang, Frank Vahid, Jun Yang, Walid Najjar

March 2005 **ACM Transactions on Architecture and Code Optimization (TACO)**, Volume 2 Issue 1

Full text available: pdf(1.32 MB) Additional Information: full citation, abstract, references, index terms

Caches contribute to much of a microprocessor system's power and energy consumption. Numerous new cache architectures, such as phased, pseudo-set-associative, way predicting, reactive-associative, way-shutdown, way-concatenating, and highly-associative, are intended to reduce power and/or energy, but they all impose some performance overhead. We have developed a new cache architecture, called a way-halting cache, that reduces energy further than previously mentioned architectures, while imposing ...

Keywords: Cache, dynamic optimization, embedded systems, low energy, low power

4 Power optimizations for cache memory: A way-halting cache for low-energy high-performance systems
Chuanjun Zhang, Frank Vahid, Jun Yang, Walid Najjar
August 2004 **Proceedings of the 2004 international symposium on Low power electronics and design**
Full text available: pdf(236.33 KB) Additional Information: full citation, abstract, references, index terms

Caches contribute to much of a microprocessor system's power and energy consumption. We have developed a new cache architecture, called a way-halting cache, that reduces energy while imposing no performance overhead. Our way-halting cache is a four-way set-associative cache that stores the four lowest-order bits of all ways' tags into a fully associative memory, which we call the halt tag array. The lookup in the halt tag array is done in parallel with, and is no slower than, the set-index decod ...

Keywords: cache design, low power techniques

5 Compilation techniques for embedded applications: Data compression for improving SPM behavior
O. Ozturk, M. Kandemir, I. Demirkiran, G. Chen, M. J. Irwin
June 2004 **Proceedings of the 41st annual conference on Design automation - Volume 00**
Full text available: pdf(145.77 KB) Additional Information: full citation, abstract, references, index terms

Scratch-pad memories (SPMs) enable fast access to time-critical data. While prior research studied both static and dynamic SPM management strategies, not being able to keep all hot data (i.e., data with high reuse) in the SPM remains the biggest problem. This paper proposes data compression to increase the number of data blocks that can be kept in the SPM. Our experiments with several embedded applications show that our compression-based SPM management heuristic is very effective and outperforms ...

Keywords: compilers, data compression, scratch-pad memory

6 A general framework for prefetch scheduling in linked data structures and its application to multi-chain prefetching
Seungryul Choi, Nicholas Kohout, Sumit Pamnani, Dongkeun Kim, Donald Yeung
May 2004 **ACM Transactions on Computer Systems (TOCS)**, Volume 22 Issue 2
Full text available: pdf(2.45 MB) Additional Information: full citation, abstract, references, index terms

Pointer-chasing applications tend to traverse composite data structures consisting of multiple independent pointer chains. While the traversal of any single pointer chain leads to the serialization of memory operations, the traversal of independent pointer chains provides a source of memory parallelism. This article investigates exploiting such *interchain memory parallelism* for the purpose of memory latency tolerance, using a technique called *multi--chain prefetching*. Previous work ...

Keywords: Data prefetching, memory parallelism, pointer-chasing code

7 A Novel Implementation of Tile-Based Address Mapping
Sambuddhi Hettiaratchi, Peter Y. K. Cheung

February 2004 **Proceedings of the conference on Design, automation and test in Europe - Volume 1**

Full text available: pdf(118.62 KB) Additional Information: full citation, abstract, index terms

Tile-based data layout has been applied to achieve various objectives such as minimizing cache conflicts and memory row switching activity. In some applications of tile-based mapping, the size of the tile can be assumed to be a power of two. In this paper, this power of two' assumption has been used to drastically simplify the tile-based address mapping functions. Once optimized, the implementation of the non-linear tile-based mapping consumes 60% less power than the implementation of the linear ...

8 Tiny instruction caches for low power embedded systems
Ann Gordon-Ross, Susan Cotterell, Frank Vahid
November 2003 **ACM Transactions on Embedded Computing Systems (TECS)**, Volume 2 Issue 4

Full text available: pdf(887.71 KB) Additional Information: full citation, abstract, references, index terms

Instruction caches have traditionally been used to improve software performance. Recently, several tiny instruction cache designs, including filter caches and dynamic loop caches, have been proposed to instead reduce software power. We propose several new tiny instruction cache designs, including preloaded loop caches, and one-level and two-level hybrid dynamic/preloaded loop caches. We evaluate the existing and proposed designs on embedded system software benchmarks from both the Powerstone and ...

Keywords: Loop cache, architecture tuning, embedded systems., filter cache, fixed program, instruction cache, low energy, low power

9 Microprocessor architecture: Frequent loop detection using efficient non-intrusive on-chip hardware
Ann Gordon-Ross, Frank Vahid
October 2003 **Proceedings of the 2003 international conference on Compilers, architecture and synthesis for embedded systems**

Full text available: pdf(278.07 KB) Additional Information: full citation, abstract, references, index terms

Dynamic software optimization methods are becoming increasingly popular for improving software performance and power. The first step in dynamic optimization consists of detecting frequently executed code, or "critical regions." Previous critical region detectors have been targeted to desktop processors. We introduce a critical region detector targeted to embedded processors, with the unique features of being very size and power efficient, and being completely non-intrusive to the software's exec ...

Keywords: dynamic optimization, frequent loop detection, frequent value profiling, hardware profiling, hot spot detection, on-chip profiling, runtime profiling

10 A pipelined memory architecture for high throughput network processors
Timothy Sherwood, George Varghese, Brad Calder
May 2003 **ACM SIGARCH Computer Architecture News , Proceedings of the 30th annual international symposium on Computer architecture**, Volume 31 Issue 2

Full text available: pdf(213.66 KB) Additional Information: full citation, abstract, references, citings

Designing ASICs for each new generation of backbone routers is a time intensive and fiscally draining process. In this paper we focus on the design of a programmable architecture for backbone routers, based on the manipulation of wide irregular memory words, that can provide a feasible design alternative to custom ASICs. We propose a pipelined memory design that emphasizes worst-case throughput over latency, and co-explore architectural tradeoffs with the design of several important network algo ...

11 Code optimization - I: Optimizing memory accesses for spatial computation
Mihai Budiu, Seth C. Goldstein
March 2003 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization**
Full text available: pdf(1.06 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

In this paper we present the internal representation and optimizations used by the CASH compiler for improving the memory parallelism of pointer-based programs. CASH uses an SSA-based representation for memory, which compactly summarizes both control-flow-and dependence information.In CASH, memory optimization is a four-step process: (1)first an initial, relatively coarse, representation of memory dependences is built; (2) next, unnecessary memory dependences are removed using dependence tests; ...

12 Energy-aware design of embedded memories: A survey of technologies, architectures, and optimization techniques
Luca Benini, Alberto Macii, Massimo Poncino
February 2003 **ACM Transactions on Embedded Computing Systems (TECS)**, Volume 2 Issue 1
Full text available: pdf(288.44 KB) Additional Information: full citation, abstract, references, index terms

Embedded systems are often designed under stringent energy consumption budgets, to limit heat generation and battery size. Since memory systems consume a significant amount of energy to store and to forward data, it is then imperative to balance power consumption and performance in memory system design. Contemporary system design focuses on the trade-off between performance and energy consumption in processing and storage units, as well as in their interconnections. Although memory design is as ...

Keywords: Embedded systems, embedded memories, integration, memories, nonvolatile, system-on-a-chip, volatile

13 Programming languages and object technologies: On optimal temporal locality of stencil codes
Claudia Leopold
March 2002 **Proceedings of the 2002 ACM symposium on Applied computing**
Full text available: pdf(433.74 KB) Additional Information: full citation, abstract, references, citings, index terms

Iterative solvers such as the Jacobi and Gauss-Seidel relaxation methods are important, but time-consuming building blocks of many scientific and engineering applications. The performance problems are largely due to cache misses, and can be reduced by tiling the codes. Whereas previous research has shown the usefulness of tiling by experimentally comparing the run times of tiled and original codes, it did not tackle the question as to whether further improvements are possible. In this paper, we ...

Keywords: data locality, lower bounds, relaxation methods, tiling

14 Distribution: Distributed component architecture for scientific applications
Roxana E. Diaconescu
February 2002 **Proceedings of the Fortieth International Confernece on Tools Pacific: Objects for internet, mobile and embedded applications - Volume 10**
Full text available: pdf(935.46 KB) Additional Information: full citation, abstract, references, index terms

The ideal goal of not having the user dealing with concurrency aspects has proven hard to achieve in the context of system (compiler, run-time) supported automatic parallelization for general purpose languages and applications. More focused approaches, of automatic parallelization for numerical applications with a regular structure have been successful. Still, they cannot fully handle irregular applications (e.g the solution of Partial Differential Equations (PDEs) for general geometries).This p ...

Keywords: concurrency, distributed components, generic programming, scientific applications

15 Architecture and Design of a High Performance SRAM for SoC Design
Shobha Singh, Shamsi Azmi, Nutan Aarawal, Penaka Phani, Ansuman Rout
January 2002 **Proceedings of the 2002 conference on Asia South Pacific design automation/VLSI Design**

Full text available: pdf(186.65 KB) Publisher Site
Additional Information: full citation, abstract

Critical issues in designing a high speed, low power static RAM in deep submicron technologies are described along with the design techniques used to overcome them. With appropriate circuit partioning, transistor sizing, choice of a suitable Sense Amplifier, a good resetting technique and judicial use of dual Vth transistors we have achieved a high speed memory without dissipating too much power. The Introduction gives the specifications of the memory that was our design target. In Section II, w ...

16 Morphable Cache Architectures: Potential Benefits
I. Kadayif, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, J. Ramanujam
August 2001 **ACM SIGPLAN Notices**, Volume 36 Issue 8

Full text available: pdf(302.99 KB)
Additional Information: full citation, abstract, references, citings, index terms

Computer architects have tried to mitigate the consequences of high memory latencies using a variety techniques. An example of these techniques is multi-level caches to counteract the latency that results from having a memory that is slower than the processor. Recent research has demonstrated that compiler optimizations that modify data layouts and restructure computation can be successful in improving memory system performance. However, in many cases, working with a fixed cache configuration ...

17 Compiler-based I/O prefetching for out-of-core applications
Angela Demke Brown, Todd C. Mowry, Orran Krieger
May 2001 **ACM Transactions on Computer Systems (TOCS)**, Volume 19 Issue 2

Full text available: pdf(499.03 KB)
Additional Information: full citation, abstract, references, citings, index terms, review

Current operating systems offer poor performance when a numeric application's working set does not fit in main memory. As a result, programmers who wish to solve "out-of-core" problems efficiently are typically faced with the onerous task of rewriting an application to use explicit I/O operations (e.g., read/write). In this paper, we propose and evaluate a fully automatic technique which liberates the programmer from this task, provides high performance, and requires only minima ...

Keywords: compiler optimization, prefetching, virtual memory

18 Session 11A: embedded tutorial: System and architecture-level power reduction of microprocessor-based communication and multi-media applications

Lode Nachtergaele, Vivek Tiwari, Nikil Dutt
November 2000 **Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design**
Full text available: pdf(34.99 KB) Additional Information: full citation, abstract, references, citings

Current microprocessor architectures become more and more dominated by the data access bottlenecks in the cache, system bus and main memory subsystems. These also have a major influence on the system (board-level) power consumption. In practice this means lower energy consumption for a given throughput requirement.In the booming domain of (largely embedded) cost-sensitive communication and multi-media applications, more and more implementations make use of microprocessor based platforms for flex ...

19 Landing CG on EARTH: a case study of fine-grained multithreading on an evolutionary path
Kevin B. Theobald, Gagan Agrawal, Rishi Kumar, Gerd Heber, Guang R. Gao, Paul Stodghill, Keshav Pingali
November 2000 **Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM)**
Full text available: pdf(150.46 KB) Publisher Site Additional Information: full citation, abstract, references, index terms

We report on our work in developing a fine-grained multithreaded solution for the communication-intensive Conjugate Gradient (CG) problem. In our recent work, we have developed a simple, yet very efficient, solution to executing matrix-vector multiply on a multithreaded system. This paper presents an effective mechanism for the reduction-broadcast phase, which is implemented and integrated with the sparse MVM resulting in a scalable implementation of the complete CG application.

20 A recursive algorithm for low-power memory partitioning
Luca Benini, Alberto Macii, Massimo Poncino
August 2000 **Proceedings of the 2000 international symposium on Low power electronics and design**
Full text available: pdf(324.58 KB) Additional Information: full citation, abstract, references, citings, index terms

Memory-processor integration offers new opportunities for reducing the energy of a system. In the case of embedded systems, one solution consists of mapping the most frequently accessed addresses onto the on-chip SRAM to guarantee power and performance efficiency. This option is especially effective when memory access patterns can be profiled and studied at design time (as in typical real-time embedded systems).In this work, we propose an algorithm for the automatic partitioning ...

Results 1 - 20 of 57 Result page: **1** 2 3 next

PORTAL
USPTO

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ● The ACM Digital Library ○ The Guide

+"memory compiler" +simulat*

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used **memory compiler simulat** Found **57** of **156,259**

Sort results by: publication date
Display results: expanded form

Save results to a Binder
Search Tips
□ Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 21 - 40 of 57 Result page: previous 1 **2** 3 next

Relevance scale

21 A compiler method for the parallel execution of irregular reductions in scalable shared memory multiprocessors
E. Gutiérrez, O. Plata, E. L. Zapata
May 2000 **Proceedings of the 14th international conference on Supercomputing**

Full text available: pdf(898.78 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a new parallelization method for reductions of arrays with subscripted subscripts on scalable shared memory multiprocessors. The mapping of computations is based on grouping reduction loop iterations into sets that are further assigned to the cooperating threads of computation. Iterations belonging to the same set are chosen in such a way that update different entries in the reduction array. That is, the loop distribution implies a conflict-free write distribution of the ...

22 Cache miss equations: a compiler framework for analyzing and tuning memory behavior
Somnath Ghosh, Margaret Martonosi, Sharad Malik
July 1999 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 21 Issue 4

Full text available: pdf(548.18 KB) Additional Information: full citation, abstract, references, citings, index terms, review

With the ever-widening performance gap between processors and main memory, cache memory, which is used to bridge this gap, is becoming more and more significant. Caches work well for programs that exhibit sufficient locality. Other programs, however, have reference patterns that fail to exploit the cache, thereby suffering heavily from high memory latency. In order to get high cache efficiency and achieve good program performance, efficient memory accessing behavior is necessary. In fact, f ...

Keywords: cache memories, compilation, optimization, program transformation

23 Transparent adaptive parallelism on NOWs using OpenMP
Alex Scherer, Honghui Lu, Thomas Gross, Willy Zwaenepoel
May 1999 **ACM SIGPLAN Notices , Proceedings of the seventh ACM SIGPLAN symposium on Principles and practice of parallel programming**, Volume 34 Issue 8

Full text available: pdf(1.26 MB) Additional Information: full citation, abstract, references, citings, index terms

We present a system that allows OpenMP programs to execute on a network of workstations with a variable number of nodes. The ability to adapt to a variable number of nodes allows a program to take advantage of additional nodes that become available after it starts execution, or to gracefully scale down when the number of available nodes is reduced. We demonstrate that the cost of adaptation is modest; the system allows a program to adapt at a moderate rate without much performance loss.Two ideas ...

24 OpenMP on networks of workstations
Honghui Lu, Y. Charlie Hu, Willy Zwaenepoel
November 1998 **Proceedings of the 1998 ACM/IEEE conference on Supercomputing (CDROM)**
Full text available: pdf(202.91 KB) Additional Information: full citation, abstract, references, citings

We describe an implementation of a sizable subset of OpenMP on networks of workstations (NOWs). By extending the availability of OpenMP to NOWs, we overcome one of its primary drawbacks compared to MPI, namely lack of portability to environments other than hardware shared memory machines. In order to support OpenMP execution on NOWs, our compiler targets a software distributed shared memory system (DSM) which provides multi-threaded execution and memory consistency.This paper presents two contri ...

25 Automatic data layout for distributed-memory machines
Ken Kennedy, Ulrich Kremer
July 1998 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 20 Issue 4
Full text available: pdf(633.20 KB) Additional Information: full citation, abstract, references, citings, index terms, review

The goal of languages like Fortran D or High Performance Fortran (HPF) is to provide a simple yet efficient machine-independent parallel programming model. After the algorithm selection, the data layout choice is the key intellectual challenge in writing an efficient program in such languages. The performance of a data layout depends on the target compilation system, the target machine, the problem size, and the number of available processors. This makes the choice of a good layout extremel ...

Keywords: high performance Fortran

26 Tolerating latency in multiprocessors through compiler-inserted prefetching
Todd C. Mowry
February 1998 **ACM Transactions on Computer Systems (TOCS)**, Volume 16 Issue 1
Full text available: pdf(410.70 KB) Additional Information: full citation, abstract, references, citings, index terms, review

The large latency of memory accesses in large-scale shared-memory multiprocessors is a key obstacle to achieving high processor utilization. Software-controlled prefetching is a technique for tolerating memory latency by explicitly executing instructions to move data close to the processor before the data are actually needed. To minimize the burden on the programmer, compiler support is needed to automatically insert prefetch instructions into the code. A key challenge when ...

Keywords: compiler optimization, prefetching

27 Using dataflow analysis techniques to reduce ownership overhead in cache coherence protocols

Jonas Skeppstedt, Per Stenström
November 1996 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 18 Issue 6

Full text available: pdf(284.68 KB) Additional Information: full citation, abstract, references, index terms, review

In this article, we explore the potential of classical dataflow analysis techniques in removing overhead in write-invalidate cache coherence protocols for shared-memory multiprocessors. We construct the compiler algorithms with varying degree of sophistication that detect loads followed by stores to the same address. Such loads are marked and constitute a hint to the cache to obtain an exclusive copy of the block so that the subsequent store does not introduce access penalties. The simplest ...

Keywords: cache coherence, dataflow analysis, performance evaluation

28 Modeling ASIC memories in VHDL
E. Balaji, P. Krishnamurthy
September 1996 **Proceedings of the conference on European design automation**

Full text available: pdf(85.09 KB) Additional Information: full citation, references, index terms

29 Compiler-directed page coloring for multiprocessors
Edouard Bugnion, Jennifer M. Anderson, Todd C. Mowry, Mendel Rosenblum, Monica S. Lam
September 1996 **Proceedings of the seventh international conference on Architectural support for programming languages and operating systems**, Volume 31 , 30 Issue 9 , 5

Full text available: pdf(1.37 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a new technique, *compiler-directed page coloring,* that eliminates conflict misses in multiprocessor applications. It enables applications to make better use of the increased aggregate cache size available in a multiprocessor. This technique uses the compiler's knowledge of the access patterns of the parallelized applications to direct the operating system's virtual memory page mapping strategy. We demonstrate that this technique can lead to significant performance impr ...

30 Improving data locality with loop transformations
Kathryn S. McKinley, Steve Carr, Chau-Wen Tseng
July 1996 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 18 Issue 4

Full text available: pdf(411.40 KB) Additional Information: full citation, abstract, references, citings, index terms, review

In the past decade, processor speed has become significantly faster than memory speed. Small, fast cache memories are designed to overcome this discrepancy, but they are only effective when programs exhibit data locality. In the this article, we present compiler optimizations to improve data locality based on a simple yet accurate cost model. The model computes both temporal and spatial reuse of cache lines to find desirable loop organizati ...

Keywords: Cache, compiler optimization, data locality, loop distribution, loop fusion, loop permutation, loop reversal, loop transformations, microprocessors, simulation

31 Compiler support for hybrid irregular accesses on multicomputers
Antonio Lain, Prithviraj Banerjee

January 1996 **Proceedings of the 10th international conference on Supercomputing**

Full text available: pdf(999.10 KB) Additional Information: full citation, references, citings, index terms

32 Compiler support for software-based cache partitioning

Frank Mueller

November 1995 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1995 workshop on Languages, compilers, & tools for real-time systems,** Volume 30 Issue 11

Full text available: pdf(886.21 KB) Additional Information: full citation, abstract, references, citings, index terms

Cache memories have become an essential part of modern processors to bridge the increasing gap between fast processors and slower main memory. Until recently, cache memories were thought to impose unpredictable execution time behavior for hard real-time systems. But recent results show that the speedup of caches can be exploited without a significant sacrifice of predictability. These results were obtained under the assumption that real-time tasks be scheduled **non-preemptively**.This paper ...

33 Compiler optimizations for eliminating barrier synchronization

Chau-Wen Tseng

August 1995 **ACM SIGPLAN Notices , Proceedings of the fifth ACM SIGPLAN symposium on Principles and practice of parallel programming,** Volume 30 Issue 8 .

Full text available: pdf(1.38 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents novel compiler optimizations for reducing synchronization overhead in compiler-parallelized scientific codes. A hybrid programming model is employed to combine the flexibility of the fork-join model with the precision and power of the single-program, multiple data (SPMD) model. By exploiting compile-time computation partitions, communication analysis can eliminate barrier synchronization or replace it with less expensive forms of synchronization. We show computation part ...

34 Data and computation transformations for multiprocessors

Jennifer M. Anderson, Saman P. Amarasinghe, Monica S. Lam

August 1995 **ACM SIGPLAN Notices , Proceedings of the fifth ACM SIGPLAN symposium on Principles and practice of parallel programming,** Volume 30 Issue 8

Full text available: pdf(1.44 MB) Additional Information: full citation, abstract, references, citings, index terms

Effective memory hierarchy utilization is critical to the performance of modern multiprocessor architectures. We have developed the first compiler system that fully automatically parallelizes sequential programs and changes the original array layouts to improve memory system performance. Our optimization algorithm consists of two steps. The first step chooses the parallelization and computation assignment such that synchronization and data sharing are minimized. The second step then restruc ...

35 Unified compilation techniques for shared and distributed address space machines

Chau-Wen Tseng, Jennifer M. Anderson, Saman P. Amarasinghe, Monica S. Lam

July 1995 **Proceedings of the 9th international conference on Supercomputing**

Full text available: pdf(1.19 MB) Additional Information: full citation, references, citings, index terms

36 Session 25: compilers: Improving the performance of DSM systems via compiler involvement

Ravi Mirchandaney, Seema Hiranandani, Ajay Sethi
November 1994 **Proceedings of the 1994 ACM/IEEE conference on Supercomputing**

Full text available: pdf(1.08 MB) Additional Information: full citation, abstract, references

Distributed shared memory (DSM) systems provide an illusion of shared memory on distributed memory systems such as workstation networks and some parallel computers such as the Cray T3D and Convex SPP-1. This illusion is provided either by enhancements to hardware, software, or a combination thereof. On these systems, users can write programs using a shared memory style of programming instead of message passing which is tedious and error prone. Our experience with one such system, TreadMarks, has ...

37 Simple compiler algorithms to reduce ownership overhead in cache coherence protocols
Jonas Skeppstedt, Per Stenström
November 1994 **Proceedings of the sixth international conference on Architectural support for programming languages and operating systems**, Volume 29 , 28 Issue 11 , 5

Full text available: pdf(1.47 MB) Additional Information: full citation, abstract, references, citings, index terms

We study in this paper the design and efficiency of compiler algorithms that remove ownership overhead in shared-memory multiprocessors with write-invalidate protocols. These algorithms detect loads followed by stores to the same address. Such loads are marked and constitute a hint to the cache to obtain an exclusive copy of the block. We consider three algorithms where the first one focuses on load-store sequences within each basic block of code and the other two analyse the existence of l ...

38 Compiler optimizations for improving data locality
Steve Carr, Kathryn S. McKinley, Chau-Wen Tseng
November 1994 **Proceedings of the sixth international conference on Architectural support for programming languages and operating systems**, Volume 29 , 28 Issue 11 , 5

Full text available: pdf(1.34 MB) Additional Information: full citation, abstract, references, citings, index terms

In the past decade, processor speed has become significantly faster than memory speed. Small, fast cache memories are designed to overcome this discrepancy, but they are only effective when programs exhibit data locality. In this paper, we present compiler optimizations to improve data locality based on a simple yet accurate cost model. The model computes both temporal and spatial reuse of cache lines to find desirable loop organizations. T ...

39 Techniques to overlap computation and communication in irregular iterative applications
Antonio Lain, Prithviraj Banerjee
July 1994 **Proceedings of the 8th international conference on Supercomputing**

Full text available: pdf(1.05 MB) Additional Information: full citation, abstract, references, citings, index terms

There are many applications in CFD and structural analysis that can be more accurately modeled using unstructured grids. Parallelization of implicit methods for unstructured grids is a difficult and important problem. This paper deals with coloring techniques to overlap computation and communication during the solution of implicit methods on message passing distributed memory multicomputers. An evaluation of coloring techniques for partitioned unstructured grids is first presented. Results ...

40 Programming, compilation, and resource management issues for multithreading (panel

session II)
Burt Halstead, David Callahan, Jack Dennis, R. S. Nikhil, Vivek Sarkar
March 1994 **ACM SIGARCH Computer Architecture News**, Volume 22 Issue 1
Full text available: pdf(1.33 MB) Additional Information: full citation, index terms

Results 21 - 40 of 57 Result page: previous 1 **2** 3 next

PORTAL
USPTO

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ● The ACM Digital Library ○ The Guide

+"memory compiler" +simulat*

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used **memory compiler simulat**

Found **57** of **156,259**

Sort results by: publication date

Display results: expanded form

Save results to a Binder

Search Tips

□ Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 41 - 57 of 57

Result page: previous 1 2 **3**

Relevance scale

41 Fortran 90D/HPF compiler for distributed memory MIMD computers: design, implementation, and performance results

Z. Bozkus, A. Choudhary, G. Fox, T. Haupt, S. Ranka

December 1993 **Proceedings of the 1993 ACM/IEEE conference on Supercomputing**

Full text available: pdf(971.18 KB) Additional Information: full citation, references, citings, index terms

42 Barrier-breaking performance for industrial problems on the CRAY C916

S. K. Graffunder

December 1993 **Proceedings of the 1993 ACM/IEEE conference on Supercomputing**

Full text available: pdf(460.76 KB) Additional Information: full citation, references, index terms

43 Runtime compilation techniques for data partitioning and communication schedule reuse

R. Ponnusamy, J. Saltz, A. Choudhary

December 1993 **Proceedings of the 1993 ACM/IEEE conference on Supercomputing**

Full text available: pdf(967.75 KB) Additional Information: full citation, references, citings, index terms

44 Cache coherence using local knowledge

E. Darnell, K. Kennedy

December 1993 **Proceedings of the 1993 ACM/IEEE conference on Supercomputing**

Full text available: pdf(1.15 MB) Additional Information: full citation, references, citings, index terms

45 Cooperative shared memory: software and hardware for scalable multiprocessors

Mark D. Hill, James R. Larus, Steven K. Reinhardt, David A. Wood

November 1993 **ACM Transactions on Computer Systems (TOCS)**, Volume 11 Issue 4

Full text available: pdf(1.37 MB) Additional Information: full citation, abstract, references, citings, index terms

We believe the paucity of massively parallel, shared-memory machines follows from the lack of a shared-memory programming performance model that can inform programmers of the cost of operations (so they can avoid expensive ones) and can tell hardware designers which cases are common (so they can build simple hardware to optimize them). Cooperative shared memory, our approach to shared-memory design, addresses this problem. Our initial implementation of cooperative shared memory u ...

Keywords: cache coherence, directory protocols, memory systems, programming model, shared-memory multiprocessors

46 PARADIGM: a compiler for automatic data distribution on multicomputers
Manish Gupta, Prithviraj Banerjee
August 1993 **Proceedings of the 7th international conference on Supercomputing**

Full text available: pdf(1.11 MB) Additional Information: full citation, abstract, references, citings, index terms

One of the most challenging steps in developing a parallel program for a distributed memory machine is determining how data should be distributed across processors. Most of the compilers being developed to make it easier to program such machines still provide no assistance to the programmer in this difficult and machine-dependent task. We have developed PARADIGM, a compiler that makes data partitioning decisions for Fortran 77 procedures. A significant feature of the design of PARADIGM is t ...

47 Speculative prefetching
Y. Jégou, O. Temam
August 1993 **Proceedings of the 7th international conference on Supercomputing**

Full text available: pdf(1.12 MB) Additional Information: full citation, abstract, references, citings, index terms

A hardware prefetching mechanism named Speculative Prefetching is proposed. This scheme detects vector accesses issued by a load/store instruction and prefetches the corresponding data. The scheme requires no software add-on, and in some cases it is more powerful than software techniques for identifying regular accesses. The tradeoffs related to its hardware implementation are extensively discussed in order to finely tune the mechanism. Experiments show that average memory ...

48 Graph contraction for physical optimization methods: a quality-cost tradeoff for mapping data on parallel computers
N. Mansour, R. Ponnusamy, A. Choudhary, G. C. Fox
August 1993 **Proceedings of the 7th international conference on Supercomputing**

Full text available: pdf(733.23 KB) Additional Information: full citation, abstract, references, citings, index terms

Mapping data to parallel computers aims at minimizing the execution time of the associated application. However, it can take an unacceptable amount of time in comparison with the execution time of the application if the size of the problem is large. In this paper, first we motivate the case for graph contraction as a means for reducing the problem size. We restrict our discussion to applications where the problem domain can be described using a graph (e.g., computational fluid dynamics appl ...

49 Cooperative shared memory: software and hardware for scalable multiprocessor
Mark D. Hill, James R. Larus, Steven K. Reinhardt, David A. Wood
September 1992 **ACM SIGPLAN Notices , Proceedings of the fifth international conference on Architectural support for programming languages and operating systems**, Volume 27 Issue 9

Full text available: Additional Information: full citation, abstract, references, citings, index pdf(1.35 MB) terms

We believe the absence of massively-parallel, shared-memory machines follows from the lack of a shared-memory programming performance model that can inform programmers of the cost of operations (so they can avoid expensive ones) and can tell hardware designers which cases are common (so they can build simple hardware to optimize them). Cooperative shared memory, our approach to shared-memory design, addresses this problem. Our initial implementation of cooperativ ...

50 Compiler and runtime support for irregularly coupled regular meshes
Craig Chase, Kay Crowley, Joel Saltz, Anthony Reeves
August 1992 **Proceedings of the 6th international conference on Supercomputing**
Full text available: pdf(1.05 MB) Additional Information: full citation, abstract, references, index terms

Regular meshes are frequently used for modeling physical phenomena on both serial and parallel computers. One advantage of regular meshes is that efficient discretization schemes can be implemented in a straightforward manner. However, geometrically-complex objects, such as aircraft, cannot be easily described using a single regular mesh. Multiple interacting regular meshes are frequently used to describe complex geometries. Each mesh models a subregion of the physical domain. The meshes, o ...

51 PYRROS: static task scheduling and code generation for message passing multiprocessors
Tao Yang, Apostolos Gerasoulis
August 1992 **Proceedings of the 6th international conference on Supercomputing**
Full text available: pdf(1.02 MB) Additional Information: full citation, abstract, references, citings, index terms

We describe a parallel programming tool for scheduling static task graphs and generating the appropriate target code for message passing MIMD architectures. The computational complexity of the system is almost linear to the size of the task graph and preliminary experiments show performance comparable to the "best" hand-written programs.

52 Architecture-independent scientific programming in data parallel C: three case studies
Philip J. Hatcher, Michael J. Quinn, Ray J. Anderson, Anthony J. Lapadula, Bradley K. Seevers, Andrew F. Bennett
August 1991 **Proceedings of the 1991 ACM/IEEE conference on Supercomputing**
Full text available: pdf(1.05 MB) Additional Information: full citation, references, citings, index terms

53 Combining hardware and software cache coherence strategies
David J. Lilja, Pen-Chung Yew
June 1991 **Proceedings of the 5th international conference on Supercomputing**
Full text available: pdf(979.07 KB) Additional Information: full citation, references, citings, index terms

54 Effective "static-graph" reorganization to improve locality in garbage-collected systems
Paul R. Wilson, Michael S. Lam, Thomas G. Moher
May 1991 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1991 conference on Programming language design and implementation**, Volume 26 Issue 6
Full text available: pdf(1.31 MB) Additional Information: full citation, references, citings, index terms

55 An architecture for software-controlled data prefetching
Alexander C. Klaiber, Henry M. Levy
April 1991 **ACM SIGARCH Computer Architecture News , Proceedings of the 18th annual international symposium on Computer architecture**, Volume 19 Issue 3
Full text available: pdf(1.16 MB) Additional Information: full citation, references, citings, index terms

56 Analysis of memory referencing behavior for design of local memories
G. D. McNiven, E. S. Davidson
May 1988 **ACM SIGARCH Computer Architecture News , Proceedings of the 15th Annual International Symposium on Computer architecture**, Volume 16 Issue 2
Full text available: pdf(845.50 KB) Additional Information: full citation, abstract, references, citings, index terms

Memory referencing behavior is analyzed via the study of traces for the purpose of developing new local memory structures and management techniques. A novel trace processing technique called flattening reduces the dependence of the results on the underlying compiler and architecture on which the trace was generated, and partitions each memory location into its constituent values. The referencing patterns of each value in the resulting trace is described via ...

57 Associative and Parallel Processors
Kenneth J. Thurber, Leon D. Wald
December 1975 **ACM Computing Surveys (CSUR)**, Volume 7 Issue 4
Full text available: pdf(2.62 MB) Additional Information: full citation, references, citings, index terms

Results 41 - 57 of 57 Result page: previous 1 2 **3**